WHAT IS CLAIMED:

1. A method of optimizing an intermediate representation of program code used during the translation of the program code, comprising:

identifying register definitions in the intermediate representation;

determining whether a top-level expression for an identified register definition is a byteswap operation; and

applying a lazy byteswapping optimization algorithm to delay performance of said byteswap operation on a value until a byteswapped value is actually required.

2. The method of claim 1, wherein the lazy byteswapping optimization algorithm comprises:

if said top-level expression is a byteswap operation, modifying the intermediate representation by:

removing said byteswap operation as the top-level expression for said identified register definition, and

wherever else in the intermediate representation where the register defined by the identified register definition is referenced, modifying the intermediate representation by inserting a byteswap operation above said referenced register in the intermediate representation;

determining whether consecutive byteswap operations are present in the modified intermediate representation; and

preventing byteswap operations appearing in the modified intermediate representation from being performed.

3. The method of claim 2, wherein consecutive byteswap operations are prevented from being performed by removing said consecutive byteswap operations from the modified intermediate representation.

4. The method of claim 2, further comprising, whenever said byteswap operation is removed as the top-level expression for said register definition, setting a lazy byteswap flag for said register definition to indicate that the value contained in the register definition is in an opposite byte order as expected.

5. The method of claim 4, wherein the intermediate representation modifying step is performed wherever else in the intermediate representation where a register having a register definition with a lazy byteswap flag that has been set is referenced.

6. The method of claim 4, further comprising clearing the set lazy byteswap flag for said referenced register when a new value is stored in said register.

7. The method of claim 6, wherein a lazy byteswap state exists that includes a set of all lazy byteswap flags for each of the registers in the intermediate representation,

further wherein each of said registers includes a respective lazy byteswap flag that is either in a set or cleared state to indicate the current state of that register.

8. The method of claim 7, further comprising a step of synchronizing the lazy byteswap state of the registers between blocks of program code being translated.

9. A computer-readable storage medium having software resident thereon in the form of computer-readable code executable by a computer to perform the following steps to optimize an intermediate representation of program code used during the translation of the program code:

identifying register definitions in the intermediate representation;

determining whether a top-level expression for an identified register definition is a byteswap operation; and

applying a lazy byteswapping optimization algorithm to delay performance of said byteswap operation on a value until a byteswapped value is actually required.

10. The computer-readable storage medium of claim 9, wherein the lazy byteswapping optimization algorithm comprises:

if said top-level expression is a byteswap operation, modifying the intermediate representation by:

removing said byteswap operation as the top-level expression for said identified register definition, and

wherever else in the intermediate representation where the register defined by the identified register definition is referenced, modifying the intermediate representation by inserting a byteswap operation above said referenced register in the intermediate representation;

determining whether consecutive byteswap operations are present in the modified intermediate representation; and

preventing byteswap operations appearing in the modified intermediate representation from being performed.

11. The computer-readable storage medium of claim 10, wherein consecutive byteswap operations are prevented from being performed by removing said consecutive byteswap operations from the modified intermediate representation.

12. The computer-readable storage medium of claim 10, said computer-readable code further executable for:

whenever said byteswap operation is removed as the top-level expression for said register definition, setting a lazy byteswap flag for said register definition to indicate that the value contained in the register definition is in an opposite byte order as expected.

13. The computer-readable storage medium of claim 12, wherein the intermediate representation modifying step is performed wherever else in the intermediate representation where a register having a register definition with a lazy byteswap flag that has been set is referenced.

14. The computer-readable storage medium of claim 12, said computer-readable code further executable for clearing the set lazy byteswap flag for said referenced register when a new value is stored in said register.

15. The computer-readable storage medium of claim 14, wherein a lazy byteswap state exists that includes a set of all lazy byteswap flags for each of the registers in the intermediate representation,

further wherein each of said registers includes a respective lazy byteswap flag that is either in a set or cleared state to indicate the current state of that register.

16. The computer-readable storage medium of claim 15, said computer-readable code further executable for synchronizing the lazy byteswap state of the registers between blocks of program code being translated.

17. An apparatus for use in a computing environment having a processor and a memory coupled to the processor for optimizing an intermediate representation of program code used during the translation of the program code, said apparatus comprising:

a register identifying mechanism for identifying register definitions in the intermediate representation;

a byteswap determining mechanism for determining whether a top-level expression for an identified register definition is a byteswap operation; and

a lazy byteswapping mechanism for applying a lazy byteswapping optimization algorithm to delay performance of said byteswap operation on a value until a byteswapped value is actually required.

18. The apparatus of claim 17, wherein the lazy byteswapping mechanism is further configured for:

if said top-level expression is a byteswap operation, modifying the intermediate representation by:

removing said byteswap operation as the top-level expression for said identified register definition, and

wherever else in the intermediate representation where the register defined by the identified register definition is referenced, modifying the intermediate representation by inserting a byteswap operation above said referenced register in the intermediate representation;

determining whether consecutive byteswap operations are present in the modified intermediate representation; and

preventing byteswap operations appearing in the modified intermediate representation from being performed.

19. The apparatus of claim 18, wherein consecutive byteswap operations are prevented from being performed by the lazy byteswapping mechanism by removing said consecutive byteswap operations from the modified intermediate representation.

20. The apparatus of claim 18, wherein the lazy byteswapping mechanism is further configured for, whenever said byteswap operation is removed as the top-level expression for said register definition, setting a lazy byteswap flag for said register definition to indicate that the value contained in the register definition is in an opposite byte order as expected.

21. The apparatus of claim 20, wherein the lazy byteswapping mechanism is further configured such that the intermediate representation is modified wherever else in the intermediate representation where a register having a register definition with a lazy byteswap flag that has been set is referenced.

22. The apparatus of claim 20, wherein the lazy byteswapping mechanism is further configured for clearing the set lazy byteswap flag for said referenced register when a new value is stored in said register.

23. The apparatus of claim 22, wherein a lazy byteswap state exists that includes a set of all lazy byteswap flags for each of the registers in the intermediate representation,

further wherein each of said registers includes a respective lazy byteswap flag that is either in a set or cleared state to indicate the current state of that register.

24. The apparatus of claim 23, further comprising a synchronizing mechanism for synchronizing the lazy byteswap state of the registers between blocks of program code being translated.